

AWS에 Jenkins+Ngnix 및 Tomcat+MySQL 서버 만들어서 빌드 및 배포하기

Jenkins 와 Ngnix 는 Amazon Linux 2 AMI 버전 지정.

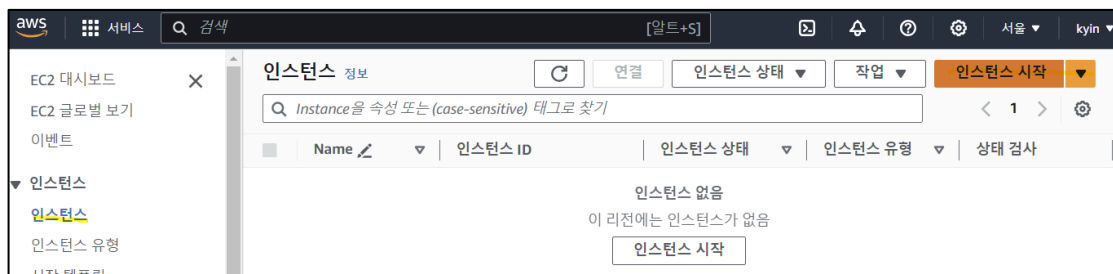
1. AWS 회원가입 및 로그인 하기

2. VPC 및 EC2 생성하기

1) 서울 리전 선택



4) 인스턴스 생성하기



이름: cicd-project-jenkins-nginx

인스턴스 시작 정보

Amazon EC2를 사용하면 AWS 클라우드에서 실행되는 가상 따라 빠르게 시작할 수 있습니다.

이름 및 태그 정보

이름

cicd-project-jenkins-nginx

Amazon Linux aws 선택 (Amazon Linux 2 AMI , 프리티어 사용 가능)

▼ Application and OS Images (Amazon Machine Image) 정보

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q 수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

Quick Start

Amazon Linux
aws

macOS
Mac

Ubuntu
ubuntu

Windows
Microsoft

Red Hat
Red Hat

SUSE L
SUSE

더 많은 AMI 찾아보기
AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine Image(AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type
ami-02d081c743d676996 (64비트(x86)) / ami-0a3c0384147c78fac (64비트(Arm))
가상화: hvm ENA enabled: true 루트 디바이스 유형: ebs

프리 티어 사용 가능

설명
Amazon Linux 2 Kernel 5.10 AMI 2.0.20240109.0 x86_64 HVM gp2

아키텍처
64비트(x86)

AMI ID
ami-02d081c743d676996

확인된 공급 업체

인스턴스 유형: t2.micro (t3.micor) (프리티어 사용 가능)

▼ 인스턴스 유형 정보 | Get advice

인스턴스 유형

t2.micro 프리 티어 사용 가능

패밀리: t2 1 vCPU 1 GiB 메모리 현재 세대: true
온디맨드 RHEL 기본 요금: 0.0744 USD 시간당
온디맨드 Linux 기본 요금: 0.0144 USD 시간당
온디맨드 SUSE 기본 요금: 0.0144 USD 시간당
온디맨드 Windows 기본 요금: 0.019 USD 시간당

모든 세대
인스턴스 유형 비교

소프트웨어가 사전 설치된 AMI에는 추가 비용이 적용됩니다.

교육센터에서 제공한 IAM 계정을 사용하는 경우에는 t2.midium (t3.midium)선택한다. (과금됨)

▼ 인스턴스 유형 정보 | 조건 받기

인스턴스 유형

t2.medium

패밀리: t2 2 vCPU 4 GiB 메모리 현재 세대: true
온디맨드 Linux 기본 요금: 0.0464 USD 시간당
온디맨드 RHEL 기본 요금: 0.0752 USD 시간당

키 페어 (로그인) 는 새로 생성하고 저장파일은 잘 보관할 것.

▼ 키 페어(로그인) 정보

키 페어를 사용하여 인스턴스에 안전하게 연결할 수 있습니다. 인스턴스를 시작하기 전에 선택한 키 페어에 대한 액세스 권한이 있는지 확인하세요.

키 페어 이름 - 필수

선택

새 키 페어 생성

키 페어 이름: cicd-project-ec2-key

키 페어 생성

키 페어 이름

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

cicd-project-ec2-key

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

키 페어 유형

☒ RSA
RSA 암호화된 프라이빗 및 퍼블릭 키 페어

☐ ED25519
ED25519 암호화된 프라이빗 및 퍼블릭 키 페어

프라이빗 키 파일 형식

☒ .pem
OpenSSH와 함께 사용

☐ .ppk
PuTTY와 함께 사용

⚠ 메시지가 표시되면 프라이빗 키를 사용자 컴퓨터의 안전하고 액세스 가능한 위치에 저장합니다. 나중에 인스턴스에 연결할 때 필요합니다. 자세히 알아보기

취소 키 페어 생성

pem 파일 저장 위치 잘 기억할 것.

← → ↕ ↗ > 내 PC > 로컬 디스크 (C:) > jenkins_work	
이름	수정한 날짜
admin_back_github	2024-01-17 오후 1:46
apache-tomcat-9.0.85	2024-01-05 오전 8:28
jenkins_boot_web	2024-01-18 오전 11:02
apache-tomcat-9.0.85.zip	2024-01-15 오전 9:39
cicd-project-ec2-key.pem	2024-01-22 오전 9:55

네트워크 설정은 다음과 같고 스토리지 구성 설정은 기본설정으로 설정한다.

▼ 네트워크 설정 정보

VPC - 필수 | 정보

vpcc-0d2d5116add3a27d9 (demo-vpc)
10.0.0.0/16

서브넷 | 정보

subnet-0d9517b6bf76c9dd6 demo-subnet-private2-us-east-1b
VPC: vpc-0d2d5116add3a27d9 소유자: 285502508151 **가용 영역: us-east-1b**
사용 가능한 IP 주소: 4091 CIDR: 10.0.144.0/20

퍼블릭 IP 자동 할당 | 정보

활성화

프리 티어 허용 범위를 벗어나는 경우 추가 요금이 적용됩니다.

방화벽(보안 그룹) | 정보

보안 그룹은 인스턴스에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 특정 트래픽이 인스턴스에 도달하도록 허용하는 규칙을 추가합니다.

☒ **보안 그룹 생성** ☐ 기존 보안 그룹 선택

보안 그룹 이름 - 필수

launch-wizard-1

이 보안 그룹은 모든 네트워크 인터페이스에 추가됩니다. 보안 그룹을 만든 후에는 이름을 편집할 수 없습니다. 최대 길이는 255자입니다. 유효한 문자는 a-z, A-Z, 0-9, 공백 및 _-./()#,@!+=&()*입니다.

설명 - 필수 | 정보

launch-wizard-1 created 2024-07-27T14:50:01.894Z

인바운드 보안 그룹 규칙

▼ 보안 그룹 규칙 1 (TCP, 22, 0.0.0.0/0)

유형 | 정보

ssh

프로토콜 | 정보

TCP

포트 범위 | 정보

22

소스 유형 | 정보

위치 무관

원본 | 정보

Q Add CIDR, prefix list or security

설명 - 선택 사항 | 정보

예: 관리자 데스크톱용 SSH

0.0.0.0/0 X

제거

마지막으로 요약 화면에서 인스턴스 시작 버튼 클릭.

▼ 요약

인스턴스 개수 | 정보

1

Virtual server type (instance type)

t2.micro

Firewall (security group)

새 보안 그룹

Storage (volumes)

1 volume(s) - 8 GiB

❗ 프리 티어: 첫 해에는 월별 프리 티어 AMI에 대한 t2.micro(또는 t2.micro를 사용할 수 없는 리전의 t3.micro) 인스턴스 사용량 750시간, EBS 스토리지 30GiB, IO 2백만 개, 스냅샷 1GB, 인터넷 대역폭 100GB가 포함됩니다.

취소 **인스턴스 시작**

다음과 같이 생성된 인스턴스 정보를 확인할 수 있다.

인스턴스 (1/1) 정보

Instance을 속성 또는 (case-sensitive) 태그로 찾기

<input checked="" type="checkbox"/>	Name ✎	인스턴스 ID	인스턴스 상태
<input checked="" type="checkbox"/>	cicd-project-jenkins-ngnix	i-0c97e187b5c93137e	✔ 실행 중

인스턴스: i-0c54bb3f478ef28c9(cicd-project-ec2)

▼ 인스턴스 요약 정보

인스턴스 ID
 i-0c54bb3f478ef28c9 (cicd-project-ec2)

IPv6 주소
-

호스트 이름 유형
IP 이름: ip-172-31-3-17.ap-northeast-2.compute.internal

퍼블릭 IPv4 주소
 3.39.228.246 | [개방 주소법](#)

인스턴스 상태
 대기 중

프라이빗 IP DNS 이름(IPv4만 해당)
 ip-172-31-3-17.ap-northeast-2.compute.internal

프라이빗 IPv4 주소
 172.31.3.17

퍼블릭 IPv4 DNS
 ec2-3-39-228-246.ap-northeast-2.compute.amazonaws.com | [개방 주소법](#)

#보안그룹에 태그 지정하여 편리하게 사용할 수 있도록 설정한다.

७|: Name

값: ccd-project-sg

[EC2](#) > [보안 그룹](#) > [sg-0bf950beac41c8114 - launch-wizard-2](#)

sg-0bf950beac41c8114 - launch-wizard-2

새부 정보

<div>보안 그룹 이름</div> <div>launch-wizard-2</div>	<div>보안 그룹 ID</div> <div>sg-0bf950beac41c8114</div>	<div>설명</div> <div>launch-wizard-2 created 2024-06-29T07:48:51.949Z</div>	<div>VPC ID</div> <div>vpc-956c6b1987f06bc76</div>
<div>소유자</div> <div>285502508151</div>	<div>인바운드 규칙 수</div> <div>1 제한 항목</div>	<div>아웃바운드 규칙 수</div> <div>1 제한 항목</div>	

인바운드 규칙

아웃바운드 규칙

태그

태그

Key	Value
아 리소스에 연결된 태그 없음	

태그 관리

태그 관리 [정보](#)

태그 관리

태그는 사용자가 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 값(선택 사항)으로 구성됩니다. 태그를 사용하여 리소스를 식별하고, 리소스에 할당된 비용을 추적할 수 있습니다.

리소스와 연결된 태그가 없습니다.

새로운 태그 추가

최대 50개의 태그(클) 더 추가할 수 있습니다.

태그 관리

태그는 사용자가 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 값(선택 사항)으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.

키

Q Name

X

값 - 선택 사항

Q cicd-project-sg

X

제거

새로운 태그 추가

최대 49개의 태그s(를) 더 추가할 수 있습니다.

취소

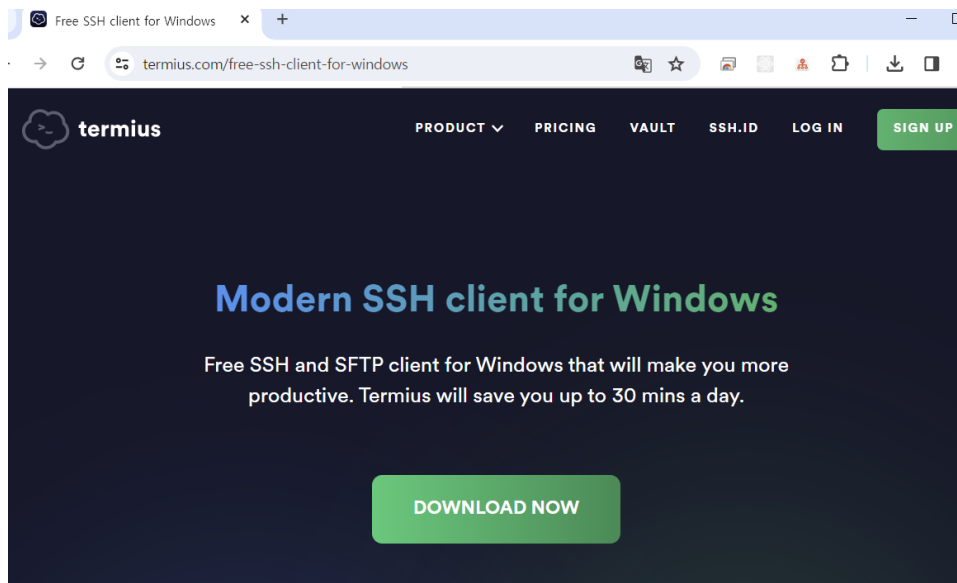
변경 사항 저장

최종적으로 다음과 같이 보안그룹의 태그가 설정된다.

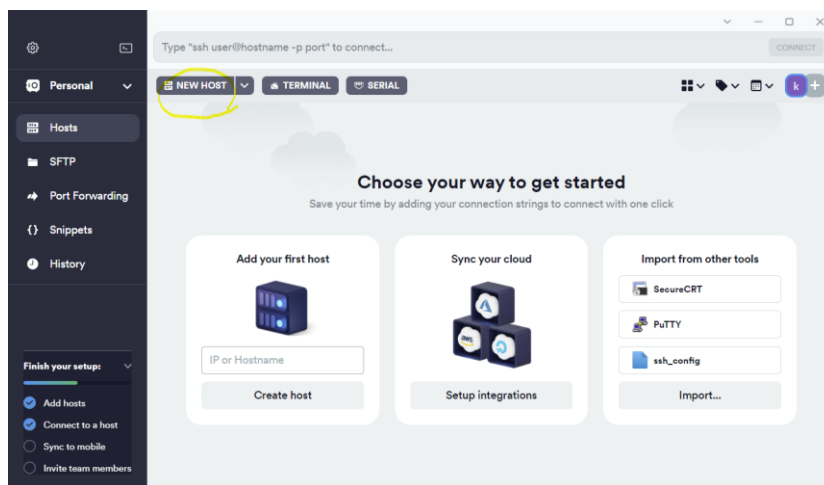
태그	
<div>Q</div>	
Key	Value
Name	cicd-project-sg

3. EC2에 접속하기 (Termius 툴 이용)

다운로드: [Termius.com/windows](https://termius.com/windows)



NEW HOST 선택



Address 에 AWS의 퍼블릭IPv4 값을 설정한다.

port번호는 기본 port인 22 사용.

Username 은 AWS의 고정 계정명인 ec2-user 사용.

Password 는 다운받은 key 이용(cicd-project-ec2-key.pem)해서 등록해야 됨.

(화면에서 Certificate 클릭하고 pem 파일명 입력하고 Create Key cicd-project-ec2-key 선택)


(AWS의 계정명은 **ec2-user** 로 고정되어 있음)

Host Details

✓ ... →

Address

AWS의 퍼블릭IPv4 주소




3.36.94.240


General

임의의 이름


AWS-EC2-jenkins



Parent Group



Tags



Backspace


Default

SSH on

22

port


AWS 고정계정명:



ec2-user

ec2-user

다운받은 key 이름 지정하고



cicd-project-ec2-key

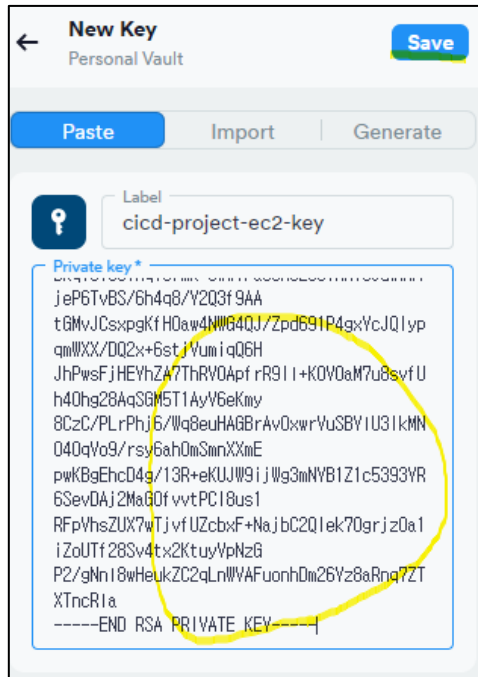
Create Key cicd-project-ec2-key

Create key 선택하기

Show more ▼

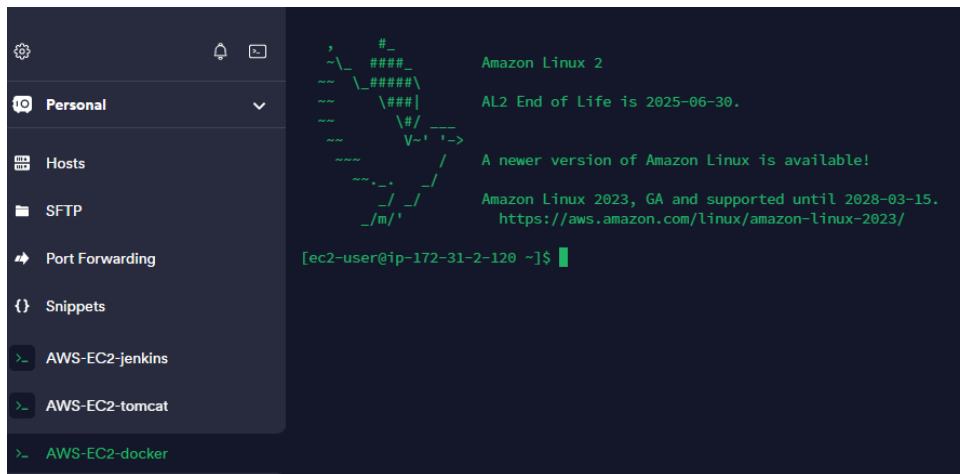
Connect

다운로드 받은 cicd-project-ec2-key.pem 파일 내용을 복사해서 붙이고 Save 버튼을 클릭한다.



마지막으로 Connect 버튼을 클릭한다.

AWS 접속이 성공하면 다음 화면이 보인다.



메모장에서 명령어를 복사해서 붙일 때는 **ctrl+shift + v** 를 이용한다.

1) EC2에 Jenkins 서버 설치하기

////////////////////////////////////

Amazon Linux 2 를 실행하는 EC2 인스턴스에 추가 라이브러리의 소프트웨어 패키지를 설치 방법.

<https://repost.aws/ko/knowledge-center/ec2-install-extras-library-software> 참조하기

\$ sudo amazon-linux-extras install epel -y

```
[ec2-user@ip-172-31-13-17 ~]$ sudo amazon-linux-extras install epel -y
Installing epel-release
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-epel amzn2extra-java-openjdk11
24 metadata files removed
10 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
amzn2extra-docker
amzn2extra-epel
amzn2extra-java-openjdk11
amzn2extra-kernel-5.10
```

\$ sudo yum update -y

////////////////////////////////////

1) JDK 11 부터 설치

JAVA 11 파일 다운로드

\$ sudo curl -L https://corretto.aws/downloads/latest/amazon-corretto-11-x64-linux-jdk.rpm -o jdk11.rpm

```
[ec2-user@ip-172-31-32-227 ~]$ sudo curl -L https://corretto.aws/downloads/latest/amazon-corretto-11-x64-linux-jdk.rpm -o jdk11.rpm
.
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left     Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:02 --:--:--   0
100 186M 100 186M    0     0 48.9M    0  0:00:03  0:00:03 --:--:-- 116M
[ec2-user@ip-172-31-32-227 ~]$ ls -l
total 190676
-rw-r--r--. 1 root root 195250751 Jan 15 08:08 jdk11.rpm
[ec2-user@ip-172-31-32-227 ~]$
```

JAVA 11 설치

\$ sudo yum localinstall jdk11.rpm

```
[ec2-user@ip-172-31-32-227 ~]$ sudo yum localinstall jdk11.rpm
Last metadata expiration check: 0:46:28 ago on Mon Jan 15 07:24:42 2024.
Dependencies resolved.
=====
Package                                Architecture      Version           Repository        Size
=====
Installing:
java-11-amazon-corretto-devel          x86_64            1:11.0.21.9-1     @commandline      186 M

Transaction Summary
=====
Install 1 Package

Total size: 186 M
Installed size: 311 M
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                                1/1
  Installing     : java-11-amazon-corretto-devel-1:11.0.21.9-1.x86_64 1/1
  Running scriptlet: java-11-amazon-corretto-devel-1:11.0.21.9-1.x86_64 1/1
  Verifying      : java-11-amazon-corretto-devel-1:11.0.21.9-1.x86_64 1/1

Installed:
java-11-amazon-corretto-devel-1:11.0.21.9-1.x86_64

Complete!
[ec2-user@ip-172-31-32-227 ~]$
```

JAVA 버전 확인

\$ javac --version

\$ java --version

```
[ec2-user@ip-172-31-32-227 ~]$ javac --version
javac 11.0.21
[ec2-user@ip-172-31-32-227 ~]$ java --version
openjdk 11.0.21 2023-10-17 LTS
OpenJDK Runtime Environment Corretto-11.0.21.9.1 (build 11.0.21+9-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.21.9.1 (build 11.0.21+9-LTS, mixed mode)
[ec2-user@ip-172-31-32-227 ~]$
```

다운받은 jdk11.rpm 삭제

\$ sudo rm -rf jdk11.rpm

```
[ec2-user@ip-172-31-32-227 ~]$ rm jdk11.rpm
rm: remove write-protected regular file 'jdk11.rpm'? y
[ec2-user@ip-172-31-32-227 ~]$ ls -l
```

Jenkins 설치

<https://pkg.jenkins.io/redhat-stable/> 참조하여 설치한다.

삭제는 다음 링크 참조:

<https://velog.io/@thovy/TROUBLESHOOTING-EC2-Linux-%EC%97%90%EC%84%9C-jenkins-%EC%A7%80%EC%9A%B0%EA%B8%B0>

Java-17-openjdk 대신에 java-11-openjdk 로 수정해서 설치한다.

Jenkins Redhat Packages

To use this repository, run the following command:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
```

If you've previously imported the key from Jenkins, the `rpm --import` will fail because you already have a

```
yum install fontconfig java-17-openjdk
yum install jenkins
```

```
$ sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
```

```
$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
```

```
$ sudo yum install fontconfig java-11-openjdk
```

```
[ec2-user@ip-172-31-3-17 ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2024-01-22 05:38:13-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.50.133, 2a04:4e42:7c::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|146.75.50.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

100%[=====>] 85 --.-K/s in 0s

2024-01-22 05:38:13 (3.27 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]
```

```
[ec2-user@ip-172-31-3-17 ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[ec2-user@ip-172-31-3-17 ~]$
```

```
[ec2-user@ip-172-31-3-17 ~]$ sudo yum install fontconfig java-11-openjdk
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No package java-11-openjdk available.
Resolving Dependencies
--> Running transaction check
---> Package fontconfig.x86_64 0:2.13.0-4.3.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

fontconfig는 설치가 되었고 다음과 같은 명령문으로 openjdk11 버전을 설치한다.

```
$ sudo amazon-linux-extras install java-openjdk11
```

```
Installed:
  fontconfig.x86_64 0:2.13.0-4.3.amzn2

Complete!

java-11-openjdk is available in Amazon Linux Extra topic "java-openjdk11"

To use, run
# sudo amazon-linux-extras install java-openjdk11 ✓
Learn more at
https://aws.amazon.com/amazon-linux-2/faqs/#Amazon_Linux_Extras

[ec2-user@ip-172-31-3-17 ~]$
```

```
[ec2-user@ip-172-31-3-17 ~]$ sudo amazon-linux-extras install java-openjdk11
Topic java-openjdk11 has end-of-support date of 2024-09-30
Installing java-11-openjdk
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-java-openjdk11 amzn2extra-
19 metadata files removed
8 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
amzn2extra-docker
```

설치된 java 확인하기

```
$ sudo /usr/sbin/alternatives --config java
```

다음 항목이 나오면 2 번을 선택하자.

```
There are 2 programs which provide 'java'.

  Selection    Command
-----
*+ 1          /usr/lib/jvm/java-11-amazon-corretto/bin/java
   2          java-11-openjdk.x86_64 (/usr/lib/jvm/java-11-openjdk-11.0.21.0.9-1.amzn2.0.1.x86_64/bin/
  java)
Enter to keep the current selection[+], or type selection number: 2
```

Jenkins 설치

```
$ sudo yum install jenkins -y
```

```

[ec2-user@ip-172-31-3-17 ~]$ sudo yum install jenkins -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.6 kB 00:00:00
Resolving Dependencies
--> Running transaction check
--> Package jenkins.noarch 0:2.426.2-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
jenkins noarch 2.426.2-1.1 jenkins 85 M

Transaction Summary
=====
Install 1 Package

Total download size: 85 M
Installed size: 85 M
Downloading packages:
jenkins-2.426.2-1.1.noarch.rpm | 85 MB 00:00:07
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : jenkins-2.426.2-1.1.noarch 1/1
Verifying : jenkins-2.426.2-1.1.noarch 1/1

Installed:
jenkins.noarch 0:2.426.2-1.1

Complete!
[ec2-user@ip-172-31-3-17 ~]$

```

Jenkins 서버에 8080 포트 접속 위한 인바운드 규칙 추가하기

인바운드 규칙 편집
정보

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

인바운드 규칙
정보

보안 그룹 규칙 ID	유형	정보	프로토콜	정보	포트 범위	소스	정보
sgr-027037454a741737b	사용자 지정 TCP	TCP	8080	사용자 지정	Q	0.0.0.0	X
sgr-01d574fe4765788c9	SSH	TCP	22	사용자 지정	Q	0.0.0.0	X
-	사용자 지정 TCP	TCP	8080	Anywhere-IPv4	Q	0.0.0.0	X

규칙 추가

인바운드 규칙 (2)
태그 관리
인바운드 규칙 편집

Q 검색
< 1 >

보안 그룹 규칙 ID	IP 버전	유형	프로토콜	포트 범위
sgr-0d361ea6b1cd405...	IPv4	사용자 지정 TCP	TCP	8080
sgr-076acfb40ffb4f8f3	IPv4	SSH	TCP	22

Jenkins 서버 시작하기

\$ `sudo systemctl status jenkins`

\$ `sudo systemctl enable jenkins`

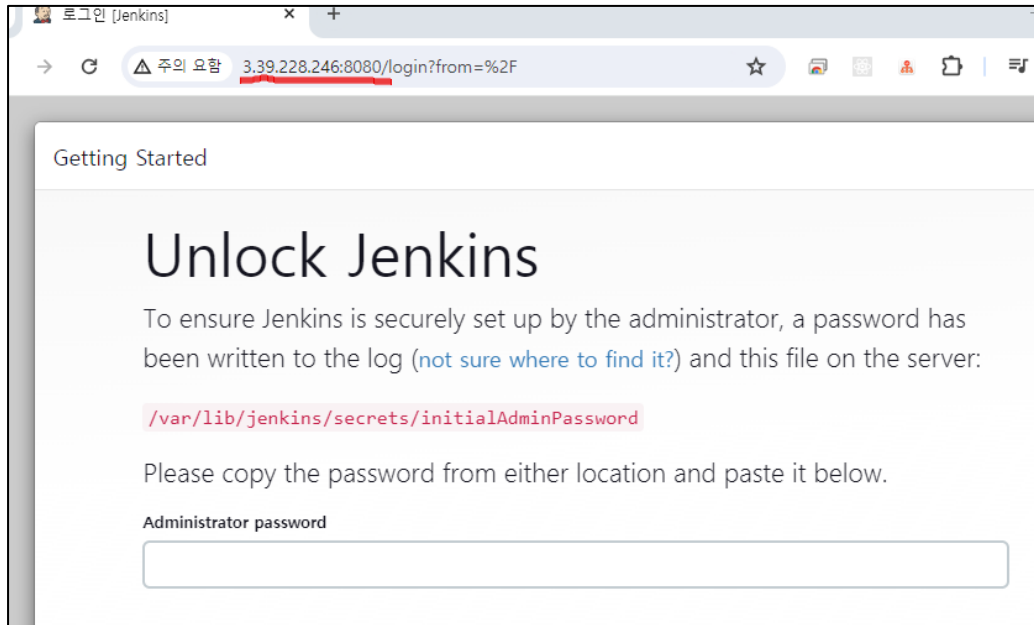
\$ `sudo systemctl start jenkins`

```
[ec2-user@ip-172-31-3-17 ~]$ sudo systemctl status jenkins
• jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
[ec2-user@ip-172-31-3-17 ~]$ sudo systemctl status jenkins
• jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
[ec2-user@ip-172-31-3-17 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-3-17 ~]$
```

웹 브라우저에서 Jenkins 서버에 요청하기

<input checked="" type="checkbox"/>	cicd-project-jenkins	i-0c54bb3f478ef28c9	실행 중	t2.micro	2/2개 검사 통
<input type="checkbox"/>	cicd-project-tomcat	i-0e81a7ffd73d3d970	실행 중	t2.micro	2/2개 검사 통
<input type="checkbox"/>	cicd-project-docker	i-02dff2a6a7548cc8f	실행 중	t2.micro	2/2개 검사 통

인스턴스: i-0c54bb3f478ef28c9(cicd-project-jenkins)		
세부 정보	Status and alarms New	모니터링 보안 네트워킹 스토리지 태그
▼ 인스턴스 요약 정보		
인스턴스 ID i-0c54bb3f478ef28c9 (cicd-project-jenkins)	퍼블릭 IPv4 주소 3.39.228.246 개방 주소법	프라이빗 IPv4 주소 172.31.3.17
IPv6 주소 -	인스턴스 상태 실행 중	퍼블릭 IPv4 DNS ec2-3-39-228-246.ap-northeast-2.compute.amazonaws.com 개방 주소법



위 경로에서 비번을 이용해서 접속한다.

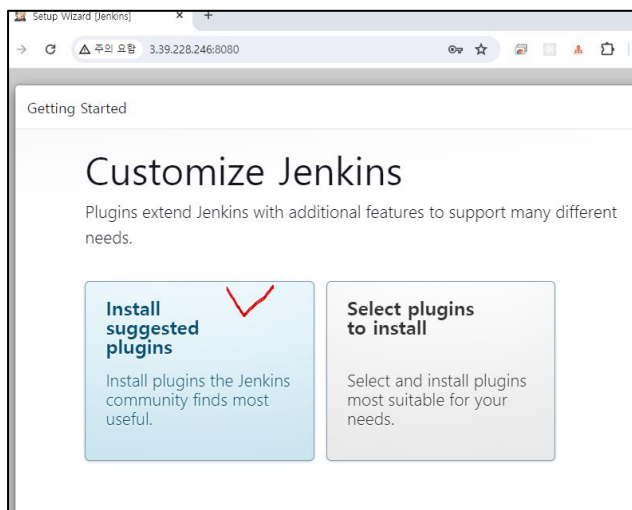
```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

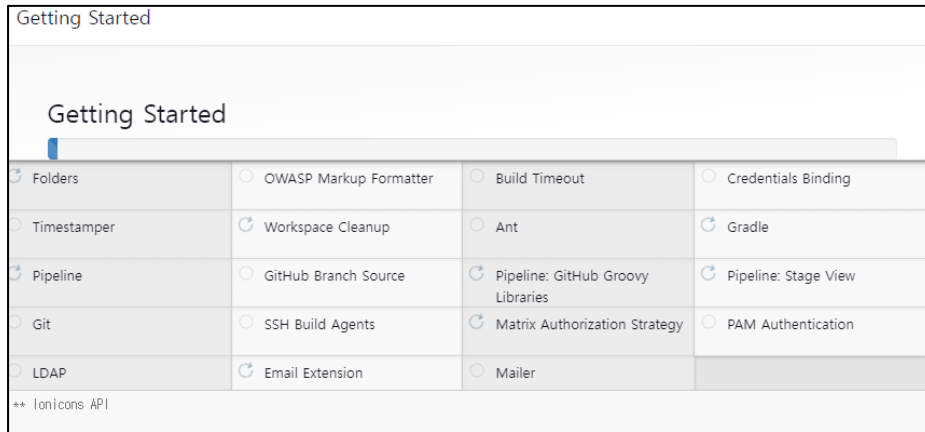
```
[ec2-user@ip-172-31-3-17 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
d4542cf8b27a474e96310bfce370a  
[ec2-user@ip-172-31-3-17 ~]$
```

터미널에서 값 복사: ctrl + shift + c

붙이기: ctrl + shift + v

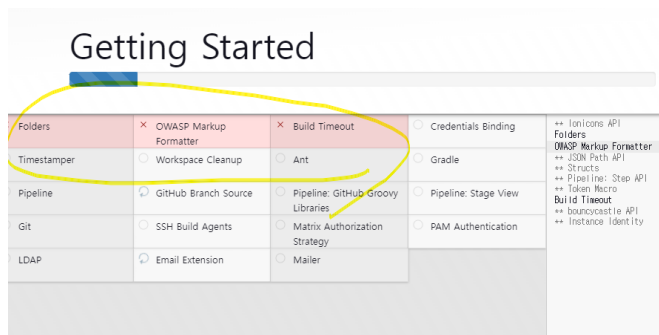
Install suggested plugins 항목을 선택한다.





////////////////////////////////////

주의: 다음 처럼 Plugins이 설치 안되는 경우가 있음. Aws 에서 모든 인스턴스 재설치할 것.



////////////////////////////////////

정상적으로 설치되면 다음과 같이 Admin 계정 생성 화면이 출력됨.

계정명: admin

암호: admin

Getting Started

Create First Admin User

계정명
admin

암호
***** admin

암호 확인

이름
Administrator

이메일 주소
hong@daum.net

Jenkins 2.426.2 [Skip and continue as admin](#) [Save and Continue](#)

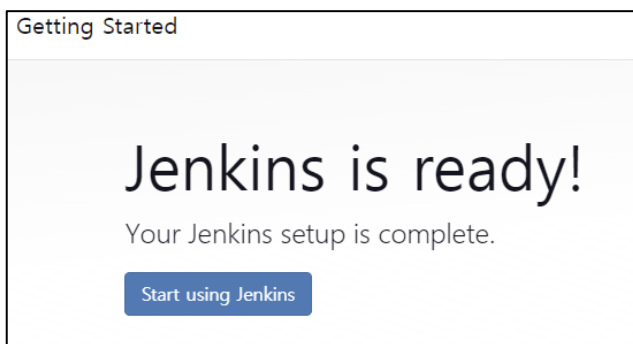
Getting Started

Instance Configuration

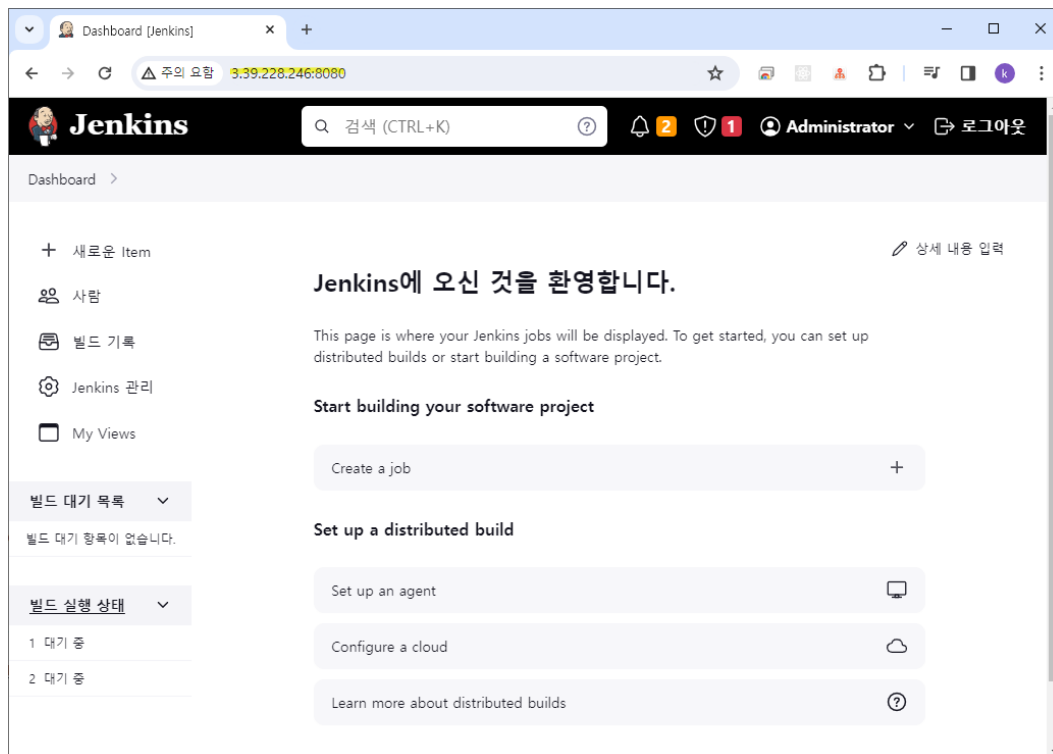
Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.



최종적으로 Jenkins 홈페이지에 접속된다.



Jenkins에 Maven 설치

```
$ cd /opt
```

반드시 maven 최신 버전 확인후 사용할 것

<https://mirror.naver.com/apache/maven/maven-3> 확인 가능

```
$ sudo wget https://mirror.naver.com/apache/maven/maven-3/3.9.6/binaries/apache-maven-3.9.6-bin.tar.gz
```

```
$ sudo tar -xvf apache-maven-3.9.6-bin.tar.gz
```

```
$ sudo mv apache-maven-3.9.6 maven
```

```
$ ls -al
```

```
$ cd maven/
```

Vi 에디터를 이용하여 다음 내용을 추가한다.

```
$ vi ~/.bash_profile
```

```
M2_HOME=/opt/maven
```

```
PATH=$PATH:$M2_HOME:$M2_HOME/bin
```

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin
M2_HOME=/opt/maven
PATH=$PATH:$M2_HOME:$M2_HOME/bin
export PATH
```

a 키 입력해서 데이터 설정하고 esc 키 입력하고 wq! 입력해서 파일을 저장한다.

```
$ source ~/.bash_profile
```

```
$ mvn --version
```

```
[ec2-user@ip-172-31-34-30 maven]$ mvn --version
Apache Maven 3.9.6 (bc0240f3c744dd6b6ec2920b3cd08dcc295161ae)
Maven home: /opt/maven
Java version: 11.0.21, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-11-openjdk-21.0.9-1.amzn2.0.1.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.10.205-195.804.amzn2.x86_64", arch: "amd64", family: "
[ec2-user@ip-172-31-34-30 maven]$
```

jenkins에 git 설치

\$ sudo yum install -y git

\$ git --version

```
[ec2-user@ip-172-31-34-30 maven]$ git --version
git version 2.40.1
[ec2-user@ip-172-31-34-30 maven]$
```

2) EC2에 Nginx 서버 설치하기

설치

```
$ sudo amazon-linux-extras install nginx1.12
```

```
[ec2-user@ip-172-31-40-247 ~]$ sudo amazon-linux-extras install nginx1.12
Topic nginx1.12 has end-of-support date of 2021-09-06
Installing nginx
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10 amzn2extra-nginx1.12
```

```
$ sudo yum update -y
```

nginx 시작

```
$ sudo systemctl status nginx
```

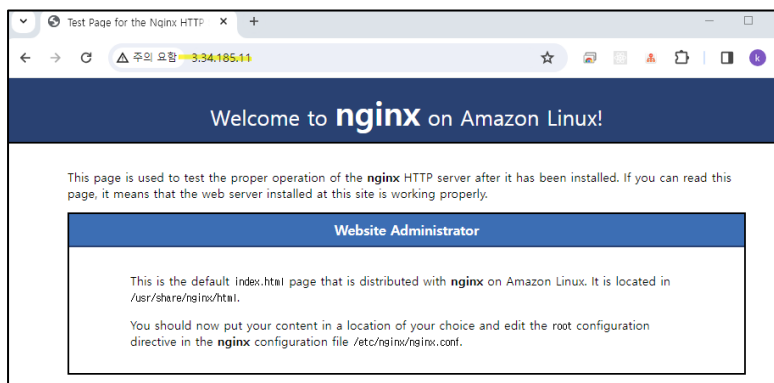
```
$ sudo systemctl enable nginx
```

```
$ sudo systemctl start nginx
```

AWS 에서 80 접근을 위한 인바운드 규칙 수정

인바운드 규칙 (2)							
Search							
<input type="checkbox"/>	Name	보안 그룹 규칙 ID	IP 버전	유형	프로토콜	포트 범위	소스
<input type="checkbox"/>	-	sgr-0d9af63d88c5e9af6	IPv4	HTTP	TCP	80	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0d86db15acf6fb92b	IPv4	SSH	TCP	22	0.0.0.0/0

nginx 서버 요청



this is the default index.html page that is distributed with nginx on Amazon Linux. It is located in /usr/share/nginx/html.

3) EC2에 tomcat 서버 설치하기

Tomcat 9 설치

opt 디렉토리 변경



\$ cd /opt











```
[ec2-user@ip-172-31-36-96 ~]$ cd /opt/  
[ec2-user@ip-172-31-36-96 opt]$
```

wget으로 다운받기

tomcat 버전이 자주 변경되기 때문에 반드시 <https://mirror.navercorp.com/apache/tomcat> 에서 최신 버전 확인하고 다운받을 것.

현재 사용 가능한 최신 버전은 다음과 같다.

Index of /apache/tomcat/tomcat-9			
Name	Last modified	Size	Description
 Parent Directory		-	
 v9.0.91/	2024-07-08 16:33	-	

Index of /apache/tomcat/tomcat-9/v9.			
Name	Last modified	Size	
 Parent Directory		-	
 embed/	2024-07-08 16:33	-	
 apache-tomcat-9.0.91-deployer.tar.gz	2024-07-02 21:43	2.8M	
 apache-tomcat-9.0.91-deployer.zip	2024-07-02 21:43	2.8M	
 apache-tomcat-9.0.91-fulldocs.tar.gz	2024-07-02 21:43	7.3M	
 apache-tomcat-9.0.91-windows-x64.zip	2024-07-02 21:43	13M	
 apache-tomcat-9.0.91-windows-x86.zip	2024-07-02 21:43	13M	
 apache-tomcat-9.0.91.exe	2024-07-02 21:43	13M	
 apache-tomcat-9.0.91.tar.gz	2024-07-02 21:43	11M	사용
 apache-tomcat-9.0.91.zip	2024-07-02 21:43	12M	

```
$ sudo wget https://mirror.navercorp.com/apache/tomcat/tomcat-9/v9.0.91/bin/apache-tomcat-9.0.91.tar.gz
```

```
[ec2-user@ip-10-0-152-97 opt]$ sudo wget https://mirror.navercorp.com/apache/tomcat/tomcat-9/v9.0.91/bin/apache-tomcat-9.0.91.tar.gz
--2024-07-27 15:34:25-- https://mirror.navercorp.com/apache/tomcat/tomcat-9/v9.0.91/bin/apache-tomcat-9.0.91.tar.gz
Resolving mirror.navercorp.com (mirror.navercorp.com)... 125.209.216.167
Connecting to mirror.navercorp.com (mirror.navercorp.com)|125.209.216.167|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11762988 (11M) [application/octet-stream]
Saving to: 'apache-tomcat-9.0.91.tar.gz'

100%[=====] 11,762,988 352KB/s in 41s

2024-07-27 15:35:07 (281 KB/s) - 'apache-tomcat-9.0.91.tar.gz' saved [11762988/11762988]

[ec2-user@ip-10-0-152-97 opt]$
```

압축풀기

```
$ sudo tar -xvzf apache-tomcat-9.0.91.tar.gz
```

```
[ec2-user@ip-10-0-152-97 opt]$ sudo tar -xvzf apache-tomcat-9.0.91.tar.gz
apache-tomcat-9.0.91/conf/
apache-tomcat-9.0.91/conf/catalina.policy
apache-tomcat-9.0.91/conf/catalina.properties
apache-tomcat-9.0.91/conf/context.xml
apache-tomcat-9.0.91/conf/jaspic-providers.xml
```

tomcat 서버의 실행권한 설정

```
$ cd apache-tomcat-9.0.91
```

```
$ sudo chmod +x ./bin/startup.sh
```

```
$ sudo chmod +x ./bin/shutdown.sh
```

```
[ec2-user@ip-172-31-8-195 opt]$ ls -l
total 20688
-rw-r--r-- 1 root root 9410508 Dec 1 2023 apache-maven-3.9.6-bin.tar.gz
drwxr-xr-x 9 root root 220 Jul 2 06:25 apache-tomcat-9.0.90
-rw-r--r-- 1 root root 11768699 Jun 14 15:01 apache-tomcat-9.0.90.tar.gz
drwxr-xr-x 4 root root 33 Jun 20 22:34 aws
drwxr-xr-x 6 root root 99 Jul 2 05:52 maven
drwxr-xr-x 2 root root 6 Aug 16 2018 rh
[ec2-user@ip-172-31-8-195 opt]$ cd apache-tomcat-9.0.90/
[ec2-user@ip-172-31-8-195 apache-tomcat-9.0.90]$ sudo chmod +x ./bin/startup.sh
[ec2-user@ip-172-31-8-195 apache-tomcat-9.0.90]$ sudo chmod +x ./bin/shutdown.sh
[ec2-user@ip-172-31-8-195 apache-tomcat-9.0.90]$
```

설정 변경

가. tomcat port 변경 (기본은 8080 임 , 8090 으로 변경)

```
$ sudo vi conf/server.xml
```

```
-->
<Connector port="8090" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443"
  maxParameterCount="1000"
  />
<!-- A "Connector" using the shared thread pool-->
```

나. webapps/manager/META-INF/context.xml 수정

```
$ sudo vi ./webapps/manager/META-INF/context.xml
```

```
-->
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
    sameSiteCookies="strict" />
  <!--Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /-->
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|N
$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
-
```

나. webapps/host-manager/META-INF/context.xml 수정

```
$ sudo vi ./webapps/host-manager/META-INF/context.xml
```

```
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
    sameSiteCookies="strict" />
  <!--Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /-->
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.apache\.c
ers\.CsrfPreventionFilter$LruCache(?:\d+)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
```

다. conf/tomcat-users.xml 권한 및 사용자 추가

```
$ sudo vi ./conf/tomcat-users.xml
```

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>

<user username="admin" password="admin" roles="manager-gui,manager-script,manager-
jmx,manager-status"/>

<user username="deployer" password="deployer" roles="manager-script"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>
```


위 코드를 복사하고 VI 에디터에서 ctrl + shift + v 하여 붙여 넣기한다.

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="admin" password="admin" roles="manager-gui,manager-script,manager-jmx,manager-status"/>
<user username="deployer" password="deployer" roles="manager-script"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>

</tomcat-users>
```

8090 으로 접속하기 위해 EC2에서 인바운드 권한 설정 추가

보안 그룹 규칙 ID	유형	정보	프로토콜	포트 범위	소스	정보	설명 - 선택 사항	정보
sgr-0e91e00b1207cda33	HTTP	TCP	80	사용자...	Q	0.0.0.0/0		삭제
sgr-02558a3d18dd3603e	SSH	TCP	22	사용자...	Q	0.0.0.0/0		삭제
sgr-073d6d53cf15f18db	사용자 지정 TCP	TCP	8080	사용자...	Q	0.0.0.0/0		삭제
-	사용자 지정 TCP	TCP	8090	Anywh...	Q	0.0.0.0/0		삭제

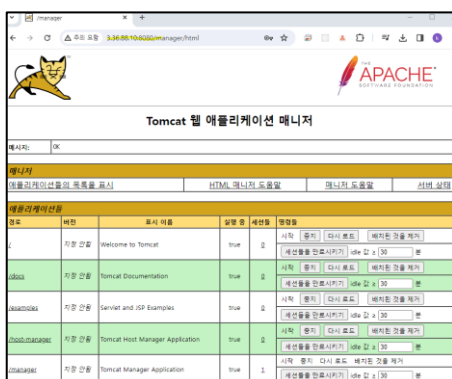
라. tomcat 서버 시작 및 요청

```
$ sudo ./bin/startup.sh
```

```
[ec2-user@ip-172-31-32-227 apache-tomcat-9.0.85]$ sudo ./bin/startup.sh
Using CATALINA_BASE:   /opt/apache-tomcat-9.0.85
Using CATALINA_HOME:   /opt/apache-tomcat-9.0.85
Using CATALINA_TMPDIR: /opt/apache-tomcat-9.0.85/temp
Using JRE_HOME:        /usr
Using CLASSPATH:        /opt/apache-tomcat-9.0.85/bin/bootstrap.jar:/opt/apache-tomcat-9.0.85/bin/tomcat-juli.jar
Tomcat started.
[ec2-user@ip-172-31-32-227 apache-tomcat-9.0.85]$
```

다음과 같이 tomcat의 관리자 창에 접속할 수 있다. (port: 8090)

http://3.36.88.10:8090/manager/html



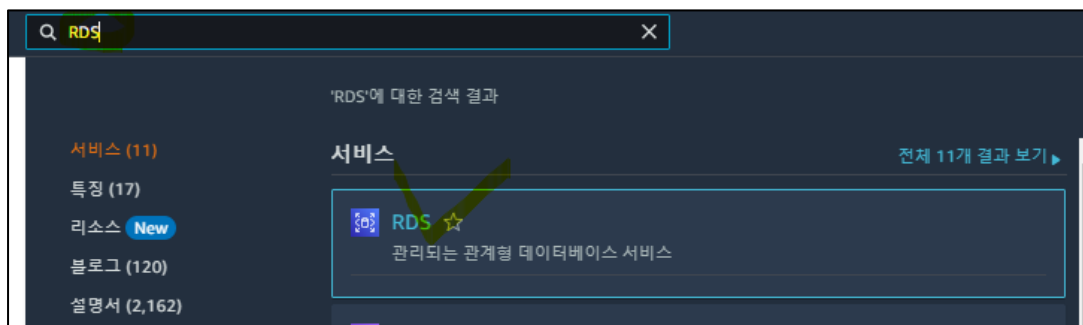
4) RDS for MySQL

가. AWS RDS 설정

testdb 데이터베이스 생성

테이블 생성

RDS 검색



데이터베이스 > 데이터베이스 생성 클릭



다음과 같이 8개의 엔진 유형이 있고 2개의 데이터베이스 생성방식이 지원됨.

데이터베이스 생성 방식은 표준 생성 선택.

데이터베이스 생성

데이터베이스 생성 방식 선택 정보

☒ 표준 생성

가용성, 보안, 백업 및 유지 관리에 대한 옵션을 포함하여 모든 구성 옵션을 설정합니다.

☐ 손쉬운 생성

권장 모범 사례 구성을 사용합니다. 일부 구성 옵션은 데이터베이스를 생성한 후 변경할 수 있습니다.

엔진 옵션은 MySQL 선택.

엔진 옵션

엔진 유형 정보

☐ Aurora (MySQL Compatible)



☐ Aurora (PostgreSQL Compatible)



☒ MySQL



☐ MariaDB



☐ PostgreSQL



☐ Oracle

ORACLE®

☐ Microsoft SQL Server



☐ IBM Db2

IBM Db2

에디션

MySQL Community

엔진 버전

정보

다음 데이터베이스 기능을 지원하는 엔진 버전을 표시합니다.

▼ 필터 숨기기

다중 AZ DB 클러스터를 지원하는 버전 표시

정보

기본 DB 인스턴스 1개와 읽기 가능한 대기 DB 인스턴스 2개로 다중 AZ DB 클러스터를 생성합니다. 다중 AZ DB 클러스터는 최대 2배 빠른 트랜잭션 커밋 지연 시간과 일반적으로 35초 미만의 자동 장애 조치를 제공합니다.

Amazon RDS 최적화된 쓰기를 지원하는 버전 표시

정보

Amazon RDS 최적화된 쓰기는 추가 비용 없이 쓰기 처리량(throughput)을 최대 2배 늘립니다.

엔진 버전

MySQL 8.0.35

RDS 확장 지원 활성화

정보

Amazon RDS 확장은 [유료 오퍼링](#)입니다. 이 옵션을 선택하면 해당 버전의 RDS 표준 지원 종료일 이후에 데이터베이스 메이저 버전을 실행하는 경우 오퍼링의 요금이 청구되는 데 동의하는 것으로 간주됩니다. [RDS for MySQL 설명서](#)에서 메이저 버전의 표준 지원 종료일을 확인하세요.

템플릿에서는 프로덕션 선택. (모든 설정이 제어 가능함.)

프로덕션 형식이지만 프리티어 설정으로 수정할 예정임

템플릿

해당 사용 사례를 충족하는 샘플 템플릿을 선택하세요.

프로덕션

고가용성 및 빠르고 일관된 성능을 위해 기본값을 사용하세요.

개발/테스트

이 인스턴스는 프로덕션 환경 외부에서 개발 용도로 마련되었습니다.

프리 티어

RDS 프리 티어를 사용하여 새로운 애플리케이션을 개발하거나, 기존 애플리케이션을 테스트하거나 Amazon RDS에서 실수 경험을 쌓을 수 있습니다. [정보](#)

가용성 및 내구성

백포 옵션

정보

아래의 백포 옵션은 위에서 선택한 엔진에서 지원하는 백포 옵션으로 제한됩니다.

다중 AZ DB 클러스터

기본 DB 인스턴스와 읽기 가능한 예비 DB 인스턴스 2개가 있는 DB 클러스터를 생성합니다. 각 DB 인스턴스는 서로 다른 가용 영역(AZ)에 있습니다. 고가용성, 데이터 이중화를 제공하고 읽기 워크로드를 처리하기 위한 용량을 늘립니다.

다중 AZ DB 인스턴스

다른 AZ에 기본 DB 인스턴스와 예비 DB 인스턴스를 생성합니다. 고가용성 및 데이터 이중화를 제공하지만 예비 DB 인스턴스는 읽기 워크로드에 대한 연결을 지원하지 않습니다.

단일 DB 인스턴스

예비 DB 인스턴스가 없는 단일 DB 인스턴스를 생성합니다.

프리티어 옵션

DB 인스턴스 식별자 정보

다음은 스토리지 설정화면으로 스토리지 자동 조정항목에 의해서 임계값을 초과하면 자동으로 EBS 볼륨을 늘린다.

스토리지

스토리지 유형 정보

이제 프로비저닝된 IOPS SSD(io2) 스토리지 볼륨을 사용할 수 있습니다.

범용 SSD(gp2)

볼륨 크기에 따라 기본 성능 결정

프리티어 옵션

할당된 스토리지 정보

20

GiB

최소값은 20GiB이고, 최대값은 6,144GiB입니다.

① 워크로드의 처리 속도를 높이기 위해 범용(SSD) 스토리지를 100GiB 미만으로 프로비저닝하여 초기 범용(SSD) I/O 크레딧 밸런스가 차감되면 오히려 지연 시간이 늘어날 수 있습니다. [자세히 알아보기](#)

② DB 인스턴스의 스토리지를 수정하면 DB 인스턴스의 상태가 스토리지 최적화 상태가 됩니다. 스토리지 최적화 작업이 완료되어도 인스턴스는 계속 사용할 수 있습니다. [자세히 알아보기](#)

▼ 스토리지 자동 조정

스토리지 자동 조정 정보

애플리케이션의 필요에 따라 데이터베이스 스토리지의 동적 조정 지원을 제공합니다.

☒ 스토리지 자동 조정 활성화
이 기능을 활성화하면 지정한 임계값 초과 후 스토리지를 늘릴 수 있습니다.

최대 스토리지 임계값 정보

데이터베이스를 지정한 임계값으로 자동 조정하면 요금이 부과됩니다.

1000

GiB

최소값은 22GiB이고, 최대값은 6,144GiB입니다.

다음은 연결 설정 화면으로 두번째 항목인 [EC2 컴퓨팅 리소스에 연결]을 선택하면 특정 EC2 인스턴스와 RDS 데이터베이스 간 연결을 자동으로 해줌.

(네트워킹 측면에서 설정이 자동으로 이루어지기 때문에 보안그룹을 다루지 않아도 됨. 따라서 초보자들에게 매우 유용함.)

하지만 실습에서는 첫 번째 항목을 선택한다. 따라서 특정 VPC에서 배포해야 된다.

데이터베이스 인증은 암호인증 으로 설정.

데이터베이스 인증

데이터베이스 인증 옵션 정보

☒ 암호 인증

데이터베이스 암호를 사용하여 인증합니다.

☐ 암호 및 IAM 데이터베이스 인증

AWS IAM 사용자 및 역할을 통해 데이터베이스 암호와 사용자 자격 증명을 사용하여 인증합니다.

☐ 암호 및 Kerberos 인증

권한이 부여된 사용자가 Kerberos 인증을 사용하여 이 DB 인스턴스에서 인증하도록 허용하려는 디렉터리를 선택합니다.

모니터링은 비활성화로 설정

모니터링

☐ 활성화된 모니터링 활성화

활성화된 모니터링 지표를 활성화하면 다른 프로세스 또는 스레드에서 CPU를 사용하는 방법을 확인하려는 경우에 유용합니다.

마지막으로 추가 구성 항목에서 데이터베이스 옵션을 설정하자.

▼ 추가 구성

데이터베이스 옵션, 암호화 켜짐, 백업 켜짐, 역추적 꺼짐, 유지 관리, CloudWatch Logs, 삭제 방지 켜짐.

데이터베이스 옵션

초기 데이터베이스 이름 정보

testddb

데이터베이스 이름을 지정하지 않으면 Amazon RDS에서 데이터베이스를 생성하지 않습니다.

DB 파라미터 그룹 정보

default.mysql8.0

옵션 그룹 정보

default:mysql-8-0

백업

☒ 자동 백업을 활성화합니다.

데이터베이스의 특정 시점 스냅샷을 생성합니다.

⚠ 자동 백업 기능은 현재 InnoDB 스토리지 엔진에 대해서만 지원됩니다. MyISAM을 사용하는 경우 [여기를](#)에서 자세한 정보를 참조하세요.

백업 보존 기간 정보

자동 백업이 유지되는 일수(1~35)입니다.

7

일

로그 내보내기

Amazon CloudWatch Logs로 게시할 로그 유형 선택

☐ 감사 로그
 ☐ 에러 로그
 ☐ 일반 로그
 ☐ 느린 쿼리 로그

IAM 역할

다음 서비스 연결 역할은 로그를 CloudWatch Logs로 게시하기 위해 사용됩니다.

RDS 서비스 연결 역할

유지 관리

자동 마이너 버전 업그레이드 정보

☒ 마이너 버전 자동 업그레이드 사용

마이너 버전 자동 업그레이드를 설정하면 새 마이너 버전이 출시되는 즉시 업그레이드됩니다. 자동 업그레이드는 데이터베이스의 유지 관리 기간 동안 수행됩니다.

유지 관리 기간 정보

보류 중인 수정 사항 또는 Amazon RDS가 데이터베이스에 적용한 유지 관리를 사용하려는 기간을 선택합니다.

☐ 기간 선택
 ☒ 기본 설정 없음

삭제 방지

☐ 삭제 방지 활성화

데이터베이스가 실수로 삭제되는 것을 방지합니다. 이 옵션을 활성화하면 데이터베이스를 삭제할 수 없습니다.

RDS > 데이터베이스

업그레이드 중 가동 중지 시간을 최소화하기 위해 블루/그린 배포 생성 고려

업그레이드하는 동안 Amazon RDS 블루/그린 배포 사용을 고려하고 가동 중지 시간을 최소화하는 것이 좋습니다. 블루/그린 배포는 프로덕션 데이터베이스 변경을 위한 스테이징 환경을 제공합니다. [RDS 사용 설명서](#) [Aurora 사용 설명서](#)

데이터베이스 (1)

그룹 리소스

수정

작업

S3에서 복원

데이터베이스 생성

데이터베이스를(를) 기준으로 필터링

DB 식별자	상태	역할	엔진	리전 및 AZ	크기	권장 사항	CPU	현재 활동
testdb	사용 가능	인스턴스	MySQL Community	ap-northeast-2b	db.t3.micro		11.04%	

인바운드 보안 규칙 3306 포트 추가설정

연결 및 보안		
엔드포인트 및 포트	네트워킹	보안
<div>엔드포인트</div> <div>database-1.cnym94h3tybq.ap-northeast-2.rds.amazonaws.com</div> <div>포트</div> <div>3306</div>	<div>가용 영역</div> <div>ap-northeast-2b</div> <div>VPC</div> <div>demo-vpc (vpc-049759aea64307d4b)</div>	<div>보안</div> <div>VPC 보안 그룹</div> <div>demo-database-mysql (sg-020285c9f79485ac1)</div> <div>활성</div> <div>퍼블릭 액세스 가능</div>

연바운드 규칙 정보

보안 그룹 규칙 ID
sgr-022a1007b5ae94960

유형 정보

MySQL/Aurora

프로토콜 정보

TCP

포트 범위 정보

3306

소스 정보

사용자 지정

실명 - 선택 사항 정보

Q

삭제

-

MySQL/Aurora

TCP

3306

Anywhere...

Q

182.222.241.225/32

X

Q

0.0.0.0/0

X

추가하기

삭제

규칙 추가

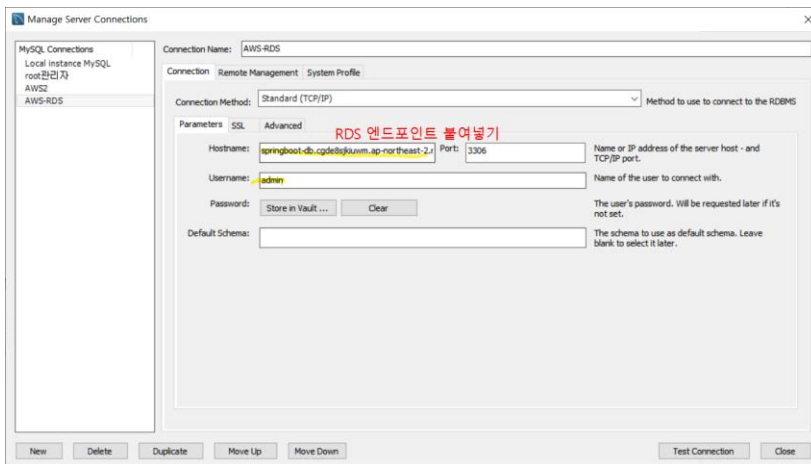
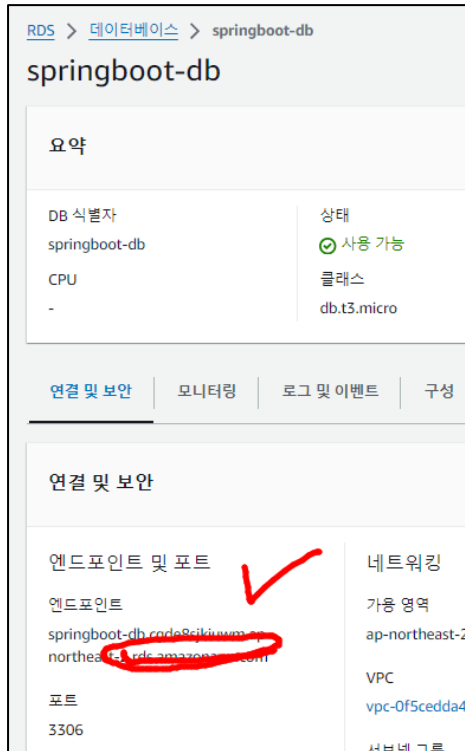
취소

변경 사항 미리 보기

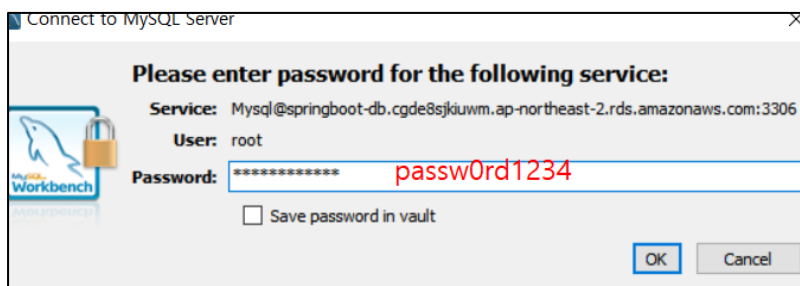
규칙 저장

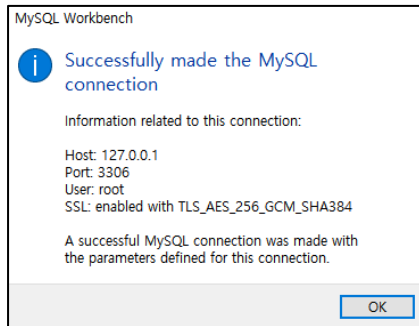
윈도우 workbench에서 접속하기

RDS 에서 엔드포인트 값을 복사



Test Connection 클릭.





테이블 생성

testdb 에 todo 및 member 테이블 생성

```
use testdb;

create table todo

( id bigint not null auto_increment COMMENT 'TODO 번호',

  userid varchar(255) not null COMMENT 'TODO 아이디',

  description varchar(255) not null COMMENT 'TODO 목록',

  targetDate date COMMENT 'TODO 목표날짜',

  done boolean COMMENT 'TODO 완료여부',

  primary key(id)

);

create table member

( userid varchar(255) not null COMMENT '아이디',

  password varchar(255) not null COMMENT '비밀번호',

  username varchar(255) not null COMMENT 'TODO 작성자',

  role varchar(255) default 'USER' not null COMMENT '역할',

  primary key(userid)

);
```

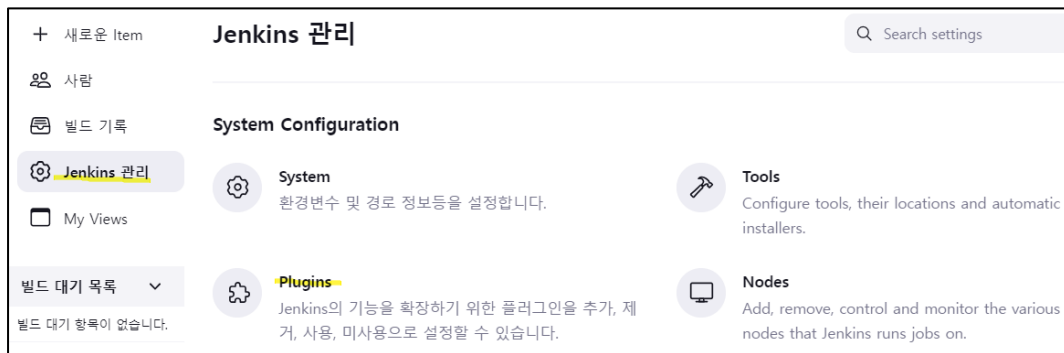
5) Jenkins에 Git과 Maven 설정하기

Github에 저장된 SpringBoot Todo 어플리케이션을 참조해서 Jenkins에 war 저장.

(명시적으로 Jenkins 에서 [지금 빌드 항목]을 선택해야 된다.)

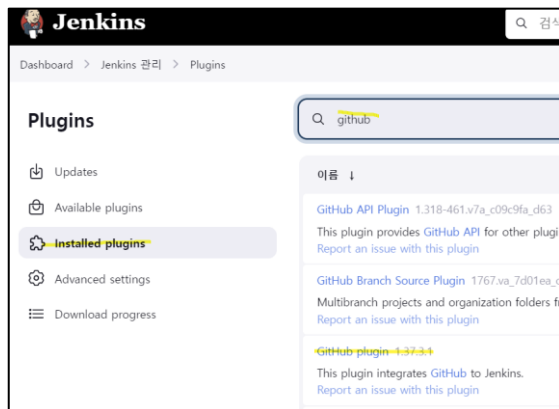
1) git plugin 추가 (Jenkins 관리 > Plugins)

과거에는 git 플러그인을 명시적으로 추가해야 했으나 현재는 자동으로 플러그인이 추가되어있음

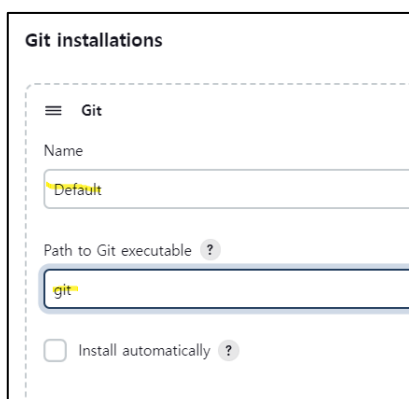


다음과 같이 github 플러그인이 미리 설치되어 있음을 확인할 수 있다.

Plugins > Installed plugins > github 로 검색



플러그인이 설치되었기 때문에 Git 을 설치할 수 있다. 현재는 설치되어 있음 (Jenkins 관리 > Tools)

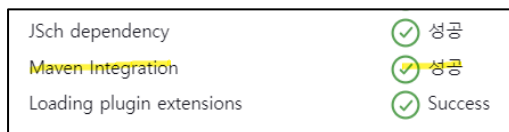


2) Maven Plugin 추가 (maven은 기본으로 설치되어 있지 않음, Jenkins 관리 > Plugins)

Available plugins > maven 검색하고 Maven Integration 선택하고 install 클릭.



설치가 성공하면 하단에 다음과 같이 Success 출력됨.

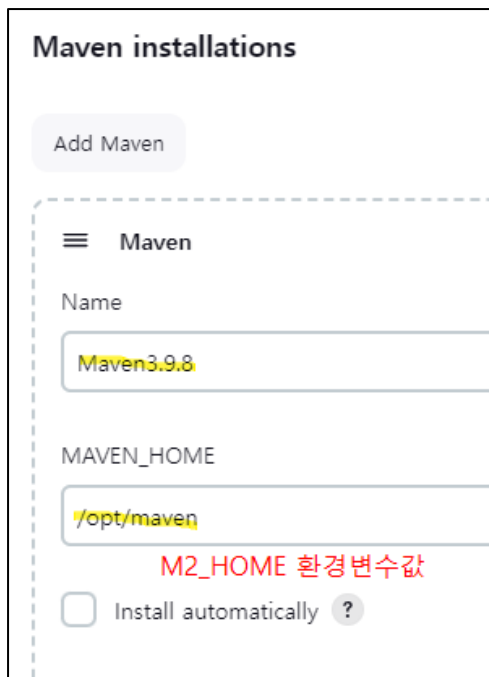


플러그인이 설치되었기 때문에 이제부터 maven 을 설치할 수 있다. (Jenkins 관리 > Tools)

다음 화면에서 Add Maven 버튼을 선택.



Name: Maven3.9.8 입력하고 Save 버튼을 선택하면 설치됨.



```
[ec2-user@ip-172-31-34-30 ~]$ echo $M2_HOME
/opt/maven
[ec2-user@ip-172-31-34-30 ~]$
```


3) Jenkins 실행단위인 Item 추가하기 (+ 새로운 Item > Todo-BootApplication 입력)


Maven을 설치했기 때문에 Maven Project를 선택한다.

Enter an item name

Todo-BootApplication Todo-BootApplication 입력

» Required field

 **Freestyle project**
Classic, general-purpose job type that checks out from up to one like archiving artifacts and sending email notifications.

 **Maven project**
Maven 프로젝트를 빌드합니다. Jenkins은 POM 파일의 이점을 가지

4) Git 저장소 지정 및 Build 설정

- SpringBoot 빌드시 web.xml 에 해당하는 코드 작성 필요.

@SpringBootApplication

```
public class Application extends SpringBootServletInitializer{  
    public static void main(String[] args) {  
        SpringApplication.run(Application.class, args);  
    }  
    @Override  
    protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {  
        return builder.sources(Application.class);  
    }  
}
```

2) application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.url=jdbc:mysql://RDB엔드포인트:3306/testdb  
?serverTimezone=UTC&characterEncoding=UTF-8  
spring.datasource.username=admin  
spring.datasource.password=Password1234
```

- pom.xml 에서 패키징 정보와 파일명 지정 및 mapper 설정 필요

```

<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.18</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.exam</groupId>
<artifactId>boot2.7_REST01_basic</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>      war 로 패키징
<name>boot2.7_REST01_basic</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>11</java.version>
</properties>

<build>
  <finalName>todo</finalName>   컨텍스트명으로 활용됨
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>true</filtering>
      <includes>
        <include>**/application*.properties</include>
      </includes>
    </resource>
    <resource>
      <directory>src/main/java</directory>
      <includes>
        <include>**/*.properties</include>
        <include>**/*.xml</include>
      </includes>
    </resource>
  </resources>      mybatis mapper 인식
</build>

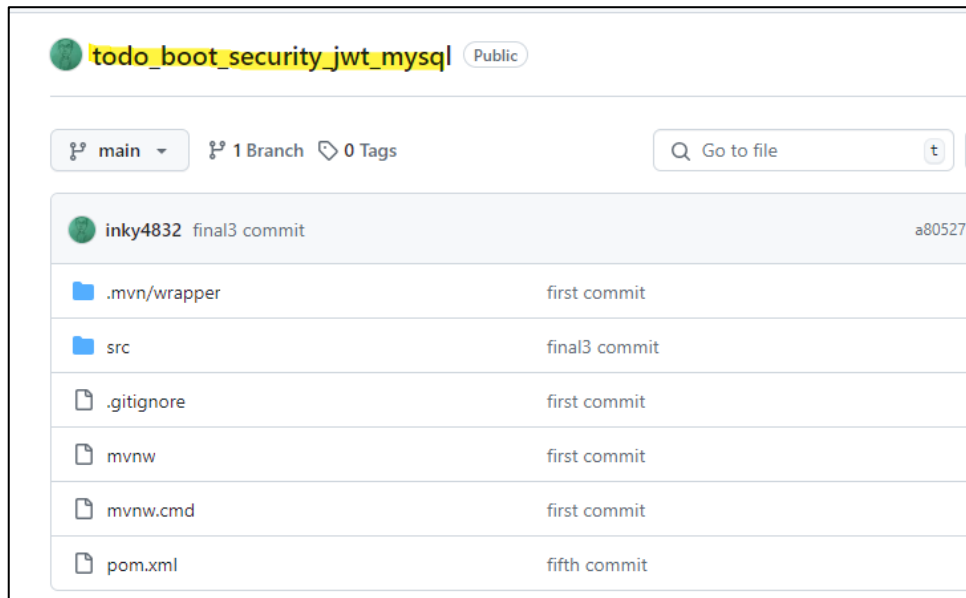
```

위 코드중에서 mybatis mapper 인식 설정은 local 환경에서는 필요 없으나 Jenkins로 war로 패키징 하여 배포시 mapper에 해당하는 xml이 패키징에서 누락되는 현상이 발생됨.

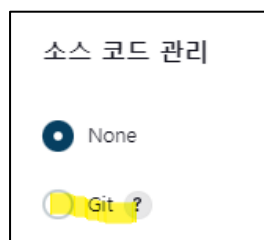
따라서 명시적으로 포함시켜야 된다.

Jenkins 에서 Github에 push된 소스코드를 배포하기 위해서 URL 경로를 지정한다.

https://github.com/inky4832/todo_boot_security_jwt_mysql.git



Todo-BootApplication > 구성(Configure) > General > 소스 코드 관리 항목



Git 체크하고 다음과 같이 설정한다.



Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

master 값을 main으로 변경

Add Branch

Repository browser ?

(자동) ▼

Additional Behaviours

Add ▼

스크롤해서 **Build > Goals and Options** 항목에 **clean compile package** 입력

Build

Root POM ?

pom.xml

Goals and options ?

clean compile package

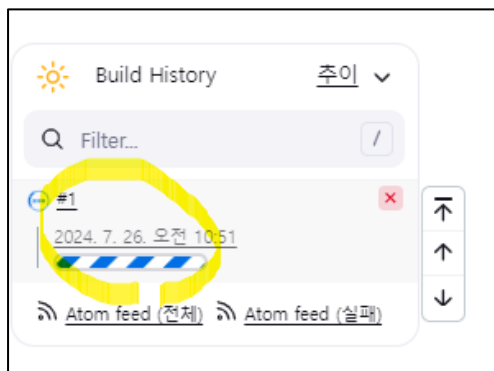
고급 ▼

[저장] 버튼 클릭.

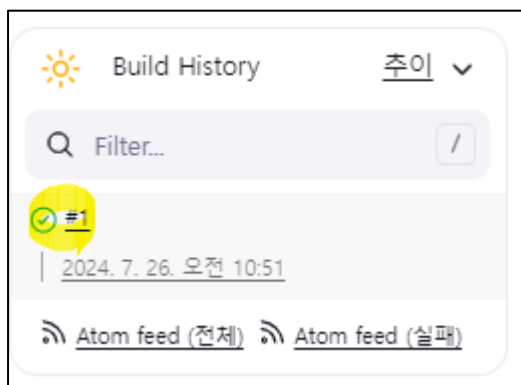
Todo-BootApplication > 지금 빌드 항목 선택하여 빌드한다.



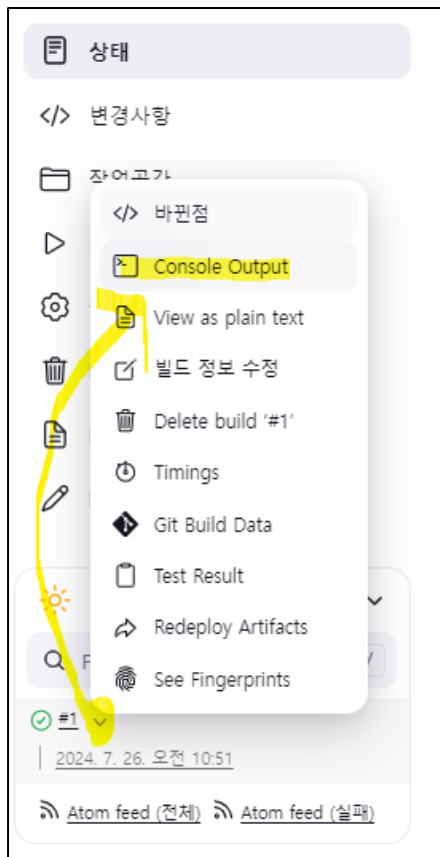
다음과 같이 빌드 중임을 알 수 있음.



이상없이 빌드되면 다음과 같은 화면이 출력됨.



실행결과 로그를 확인하기 위하여 Console Output 선택.



다음과 같이 todo.war 파일이 생성되고 SUCCESS 출력이 된다.

```
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 52.876 s
[INFO] Finished at: 2024-07-28T14:04:12Z
[INFO] -----
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/ToDo-BootApp/pom.xml to
com.exam/boot2.7_REST01_basic/0.0.1-SNAPSHOT/boot2.7_REST01_basic-0.0.1-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/ToDo-BootApp/target/todo.war to
com.exam/boot2.7_REST01_basic/0.0.1-SNAPSHOT/boot2.7_REST01_basic-0.0.1-SNAPSHOT.war
channel stopped
Finished: SUCCESS
```

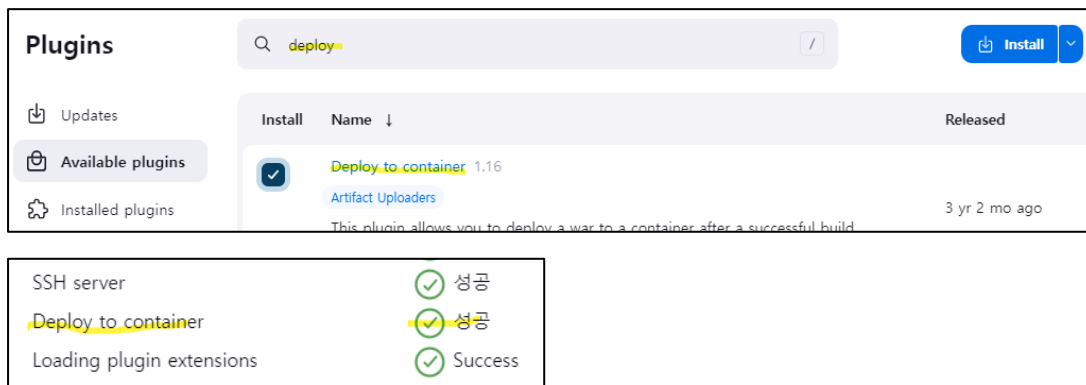
지금까지의 실습 결과는 Github에 push된 Todo 어플리케이션을 Jenkins에서 접근해서 Jenkins가 설치된 PC에 todo.war로 패키징 되었음.

5. Jenkins에 Tomcat9 설정하기

Github에 저장된 웹 어플리케이션을 참조해서 Jenkins에 told.war로 저장하고 사용중인 Tomcat9 서버에 배포까지 처리. (명시적으로 Jenkins 에서 [지금 빌드 항목]을 선택해야 된다.)

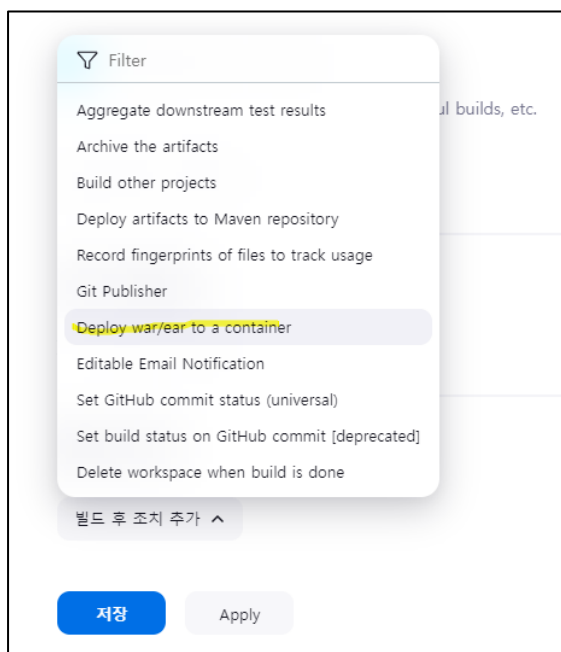
1) Tomcat Plugin 추가 (Jenkins 관리 > Plugins)

Available plugins 에서 Deploy 검색어 입력하고 Deploy to container 플러그인 설치하기



2) 빌드 후 조치 (현재 진행중인 Todo-BootApplication item 에서 설정)

Todo-BootApplication > 구성(Configuration) > General > 빌드 후 조치 > 빌드 후 조치 추가 선택 > Deploy war/ear to a container 선택



WAR/EAR files 항목에 ****/*.war** 입력하고 Context Path 항목은 비워둔다.

빌드 후 조치

Deploy war/ear to a container

WAR/EAR files ?

****/*.war** ****/*.war**

Context path ?

Containers

Add Container ▾

☐ Deploy on failure

Add Container 클릭 > Tomcat 9.x Remote 선택하기

Build 후 조치

Filter

- GlassFish 2.x
- GlassFish 3.x
- GlassFish 4.x
- JBoss AS 3.x
- JBoss AS 4.x
- JBoss AS 5.x
- JBoss AS 6.x
- JBoss AS 7.x
- Tomcat 4.x Remote
- Tomcat 5.x Remote
- Tomcat 6.x Remote
- Tomcat 7.x Remote
- Tomcat 8.x Remote
- Tomcat 9.x Remote**

Add Container ⬆

+ Add 클릭 > Jenkins 선택하기

The screenshot shows the 'Build After' configuration in Jenkins. The 'Deploy war/ear to a container' plugin is selected. The 'WAR/EAR files' field contains '**/*war'. The 'Context path' field is empty. Under 'Containers', 'Tomcat 9.x Remote' is selected. The 'Credentials' dropdown shows '- none -'. An 'Add' button is highlighted, and a tooltip for 'Jenkins' is visible.

Tomcat 9 에 접근해서 배포하기 위한 권한을 설정해야 된다.

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>

<user username="admin" password="admin"
      roles="manager-gui,manager-script,manager-jmx,manager-status"/>
<user username="deployer" password="deployer" roles="manager-script"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>
```

The screenshot shows the 'Jenkins Credentials Provider: Jenkins' configuration. The 'Domain' is 'Global credentials (unrestricted)'. The 'Kind' is 'Username with password'. The 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'deployer' and 'deployer'. The 'Treat username as secret' checkbox is unchecked. The 'Password' field contains 'deployer' and 'deployer'.

Credentials 항목에는 `deployer/*****` 선택하고 Tomcat URL 에는 <http://aws퍼블릭ip:8090/> 입력한다.

최종 설정 화면은 다음과 같다.

Deploy war/ear to a container

WAR/EAR files ?

Context path ?

Containers

Tomcat 9.x Remote

Credentials

deployer/*****

+ Add

Tomcat URL ? **http://aws퍼블릭ip:8090/**

http://43.203.117.70:8090/

고급

Add Container

저장하고 지금 빌드 항목 선택.

(빌드하기전에 반드시 Tomcat 이 실행되어 있어야 됨)

Build History

추이

Filter builds...

#2

2024. 1. 15. 오후 3:32

#1

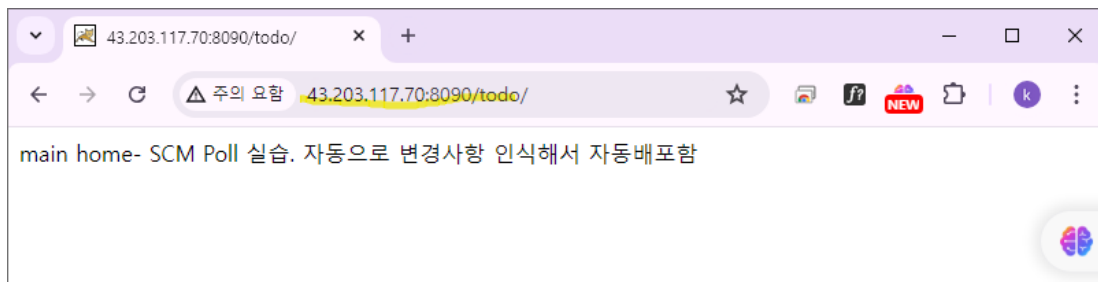
2024. 1. 15. 오후 3:06

Atom feed (전체) Atom feed (실패)

빌드가 성공하면 다음과 같이 Tomcat 의 webapps에 todo.war 파일이 배포된다.

/host-manager	지정 안됨	Tomcat Host Manager Application	true	0	<div>시작 중지 다시 로드 배치된 것을 제거</div> <div>세션들을 만료시키기 idle 값 ≥ 30 분</div>
/manager	지정 안됨	Tomcat Manager Application	true	1	<div>시작 중지 다시 로드 배치된 것을 제거</div> <div>세션들을 만료시키기 idle 값 ≥ 30 분</div>
/todo	지정 안됨		true	0	<div>시작 중지 다시 로드 배치된 것을 제거</div> <div>세션들을 만료시키기 idle 값 ≥ 30 분</div>

웹 브라우저에서 http://43.203.117.70:8090/todo/ 요청 화면은 다음과 같다.



6. Jenkins에 Poll SCM 설정하기

Github에 새롭게 commit 된 변경사항이 발생될 때 마다 자동으로 저장된 웹 어플리케이션을 참조해서 Jenkins에 war 저장하고 사용중인 Tomcat 서버에 배포까지 처리.

1) 빌드 유발 항목에서 Setup Poll SCM 활성화

(Todo-BootApplication > 구성 > 빌드 유발

빌드 유발

- ☒ Build whenever a SNAPSHOT dependency is built ?
- ☐ Schedule build when some upstream has no successful builds ?
- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub hook trigger for GITScm polling ?
- ☒ Poll SCM ?

Schedule ?

`* * * * *`

⚠ Do you really mean "every minute" when you say `"* * * * *"`? Perhaps you meant `"H * * * *"` to poll once per hour
Would last have run at 2024년 7월 26일 금요일 오후 4시 19분 20초 대한민국 표준시; would next run at 2024년 7월 26일 금요일 오후 4시 19분 20초 대한민국 표준시.

☐ Ignore post-commit hooks ?

`*(분, 0~59) *(시, 0~23) *(일, 1~31) *(월, 1~12) *(요일, 0~6)`

7. ReactJS 프론트엔드 어플리케이션 배포

먼저 localhost 로 되어 있는 URL 값을 모두 AWS 퍼블릭 IP로 변경해야 된다.

5) nginx 서버에 Reactjs 어플리케이션을 배포하기

- VSC에서 reactjs 어플리케이션을 빌드하자.

npm run build 명령어 사용

```
C:\reactjs_study\my-app>npm run build

> my-app@0.1.0 build
> react-scripts build

Creating an optimized production build...
█
```

File sizes after gzip:

```
68.32 kB  build\static\js\main.138119b0.js
31.96 kB  build\static\css\main.a1674e4c.css
1.77 kB   build\static\js\453.ed3810f9.chunk.js
```

The project was built assuming it is hosted at `/`.
You can control this with the `homepage` field in your `package.json`.

The `build` folder is ready to be deployed.
You may serve it with a static server:

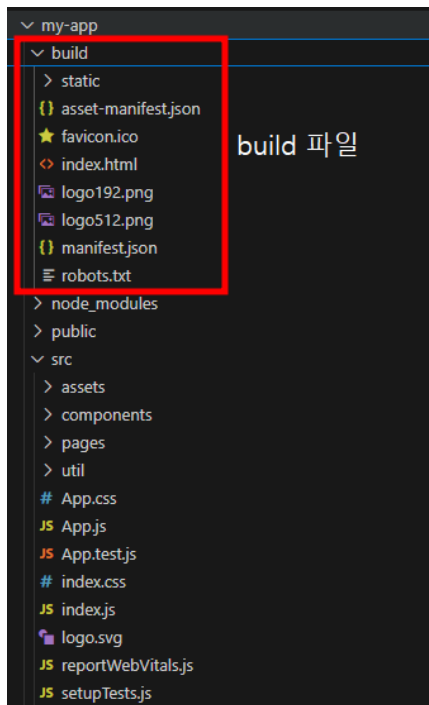
```
npm install -g serve
serve -s build
```

Find out more about deployment here:

<https://cra.link/deployment>

```
C:\reactjs_study\my-app>
```

build 명령어로 생성된 파일들은 다음과 같다.



7) Github (todo_react 원격 저장소)에 build 폴더 Push 하기

```
$ cd build
```

```
$ git init
```

```
$ git config --global core.autocrlf true
```

```
$ git add .
```

commit 메시지는 반드시 "" (쌍따옴표) 사용할 것

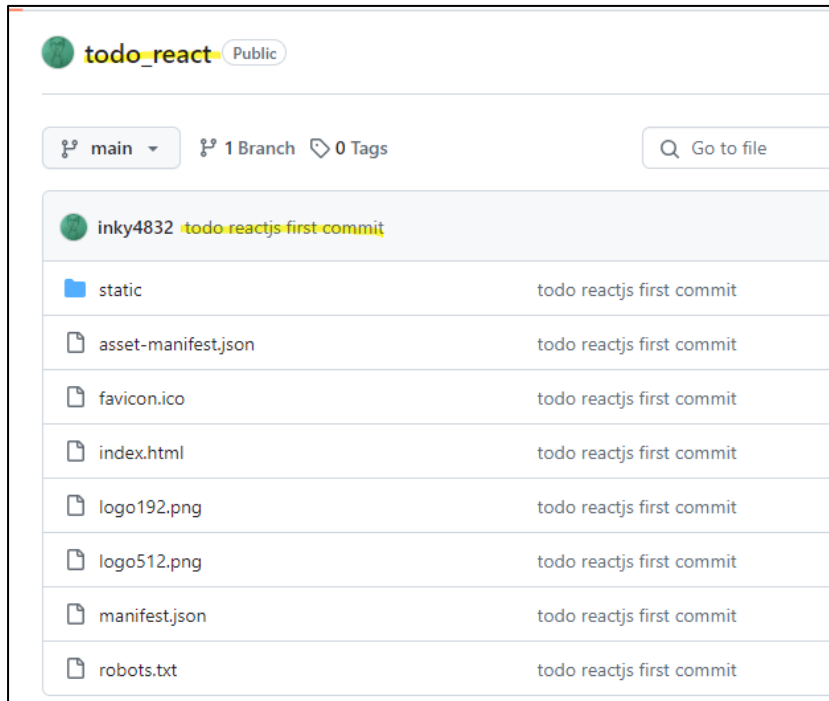
```
$ git commit -m "todo reactjs first commit"
```

```
$ git remote add origin https://github.com/inky4832/todo_react.git
```

```
$ git branch -M main
```

```
$ git push -u origin main    ( +main 지정하면 강제로 push )
```

성공적으로 github에 push 하면 다음과 같이 todo_react 원격 저장소에 push된 파일들을 확인가능.



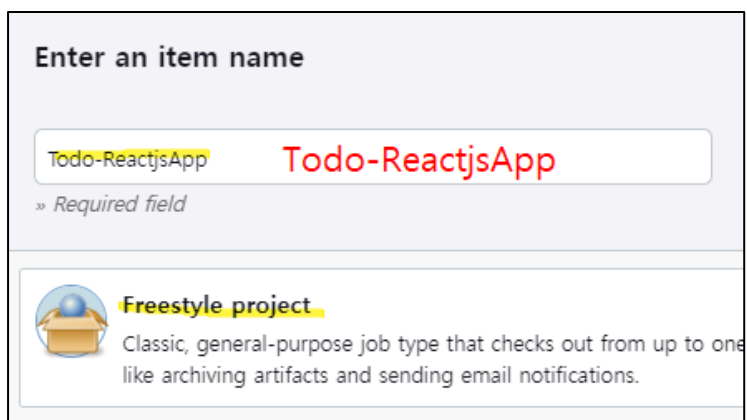
8) Jenkins 에서 Reactjs 어플리케이션 배포

AWS-EC2-jenkins-nginx 에서 다음 작업 실행

```
$ sudo chmod 777 /usr/share/nginx/html
```

가. 새로운 item 생성하기

Todo-ReactjsApp 이름의 Freestyle project 설정



나. 소스 코드 관리

Todo-ReactjsApp > 구성 > 소스 코드 관리

https://github.com/inky4832/todo_react.git

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/inky4832/todo_react.git

Credentials ?

- none -

+ Add ▾

고급 ▾

Add Repository

Branches to build ?

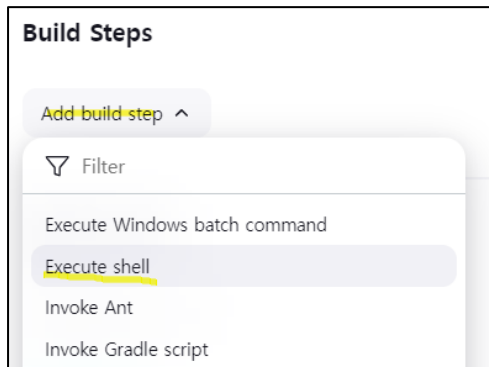
Branch Specifier (blank for 'any') ?

[*/main](#) [*/main](#)

Add Branch

다. Build steps

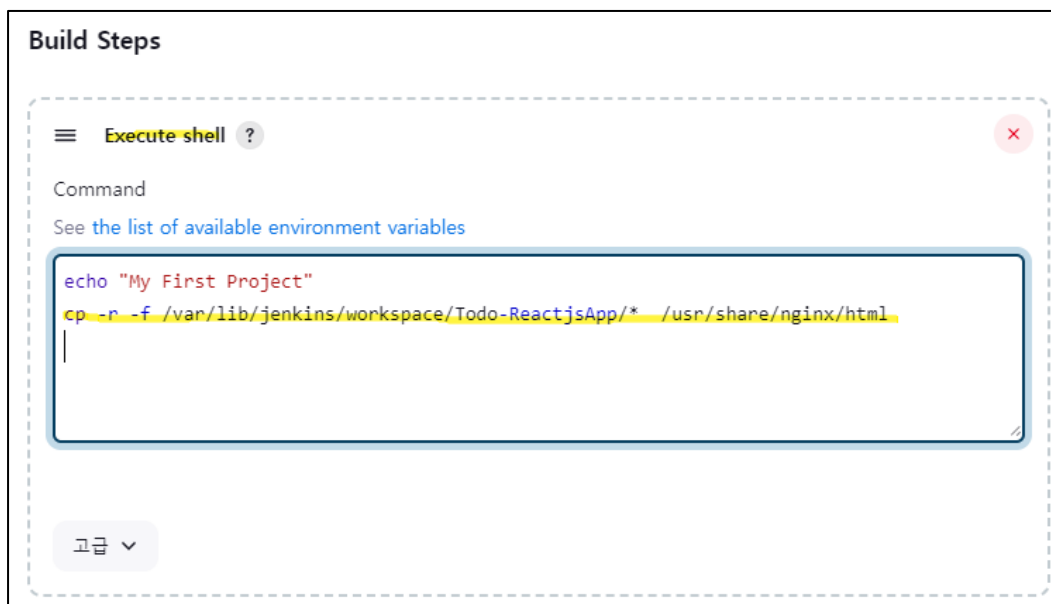
Execute shell 선택



Command 항목에 다음 내용을 기입한다.

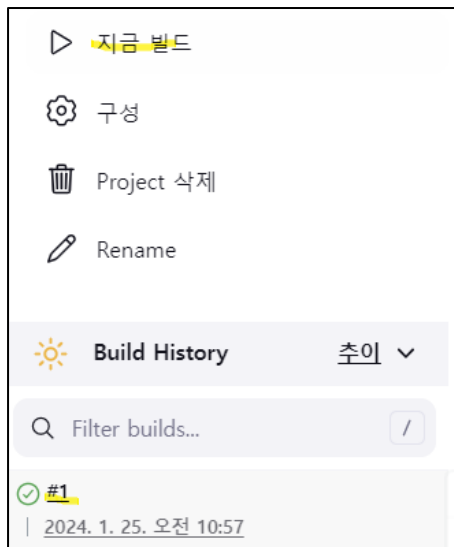
```
echo "My First Project"
```

```
cp -r -f /var/lib/jenkins/workspace/ToDo-ReactjsApp/* /usr/share/nginx/html
```



[저장] 버튼 클릭

라. 지금 빌드 하기

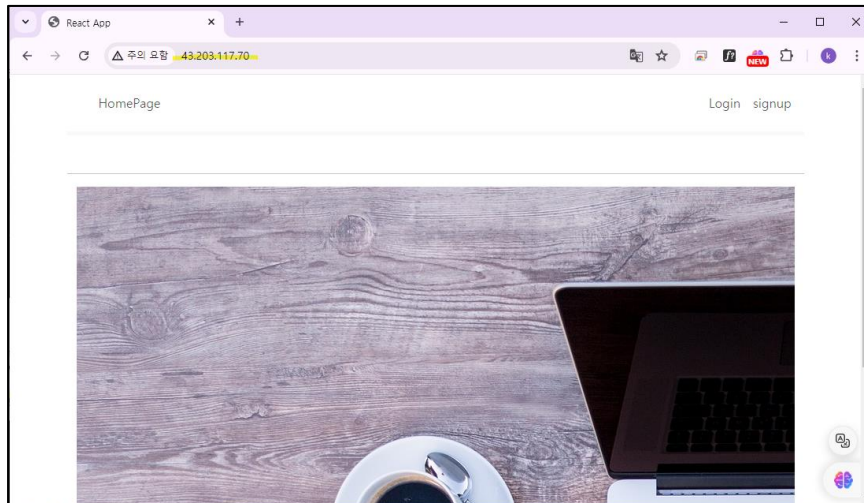


마. 복사된 파일 확인하기

\$ `cd /usr/share/nginx/html`

```
[ec2-user@ip-10-0-22-79 apache-tomcat-9.0.91]$ cd /usr/share/nginx/html/
[ec2-user@ip-10-0-22-79 html]$ pwd
/usr/share/nginx/html
[ec2-user@ip-10-0-22-79 html]$ ls -al
total 48
drwxrwxrwx 3 root    root      193 Jul 28 14:32 .
drwxr-xr-x 4 root    root        33 Jul 28 13:16 ..
-rw-r--r-- 1 root    root    3665 Aug 28  2019 404.html
-rw-r--r-- 1 root    root    3708 Aug 28  2019 50x.html
-rw-r--r-- 1 jenkins jenkins  617 Jul 28 14:32 asset-manifest.json
-rw-r--r-- 1 jenkins jenkins 3870 Jul 28 14:32 favicon.ico
-rw-r--r-- 1 jenkins jenkins  666 Jul 28 14:32 index.html
-rw-r--r-- 1 jenkins jenkins 5347 Jul 28 14:32 logo192.png
-rw-r--r-- 1 jenkins jenkins 9664 Jul 28 14:32 logo512.png
-rw-r--r-- 1 jenkins jenkins  492 Jul 28 14:32 manifest.json
-rw-r--r-- 1 jenkins jenkins   67 Jul 28 14:32 robots.txt
drwxr-xr-x 5 jenkins jenkins   40 Jul 28 14:32 static
[ec2-user@ip-10-0-22-79 html]$
```

바. nginx 서버 요청하기



자. 빌드 유발 항목에서 Setup Poll SCM 활성화

(Todo-ReactjsApp > 구성 > 빌드 유발

빌드 유발

☒ Build whenever a SNAPSHOT dependency is built ?

☐ Schedule build when some upstream has no successful builds ?

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☐ GitHub hook trigger for GITScm polling ?

☒ **Poll SCM** ?

Schedule ?

* * * * *

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * * *" to poll once per hour
Would last have run at 2024년 7월 26일 금요일 오후 4시 19분 20초 대한민국 표준시; would next run at 2024년 7월 26일 금요일 오후 4시 19분 20초 대한민국 표준시.

☐ Ignore post-commit hooks ?

* (분, 0~59) * (시, 0~23) * (일, 1~31) * (월, 1~12) * (요일, 0~6)

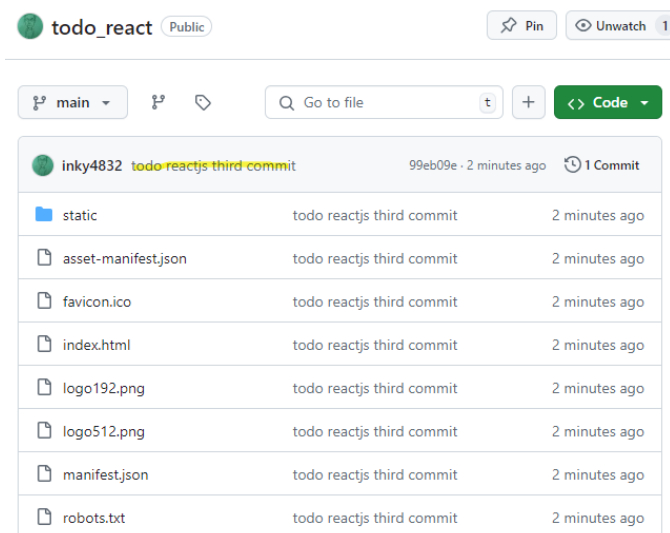
차. VSC 에서 코드 수정하고 변경사항 자동으로 배포하기

- 메인 페이지의 홈페이지 링크를 HomePage 링크로 변경.
- npm run build (**my-app** 디렉터리에서 실행)
- git init (**build** 디렉터리에서 실행)

- `git config --global core.autocrlf true`
- `git add`
- `git commit -m "todo reactjs third commit"`
- `git remote add origin https://github.com/inky4832/todo_react.git`
`git push -u origin +main` (+main 지정하면 강제 push 됨)

Jenkins 에서 자동으로 github의 변경 사항을 체크해서 자동으로 배포해줌.

github 화면



nginx 서버 요청₩

다음과 같이 HomePage 링크로 변경된 것을 확인 할 수 있음.

